Analysis of delay patterns and correlations in railway traffic data

Roland Krisztián Szabó, Tomáš Horváth, and Ádám Tarcsi

Eötvös Loránd University, Faculty of Informatics, Budapest, Pázmány Péter stny. 1/C., 1117,
rolandszabo@inf.elte.hu tomas.horvath@inf.elte.hu ade@inf.elte.hu

Abstract: Traffic itself can be a huge challenge for most commuters regardless of the transportation method of their choice. For example, it is inevitable to experience delays and congestion during rush hours. All commute methods have their own specific characteristics when it comes to delays - cars and buses suffer from traffic jams and similar principles apply to railways as well. However, the causes of railway delays are not that straightforward and they need further investigation. According to our personal experiences most passengers are not aware of the reasons behind train delays even though they are usually encountered multiple times a day. In this paper I will present possible answers based on the data collected from the publicly available APIs of Hungarian State Railways over the past 1.5 years.

1 Datasets

The idea of the delay analysis and prediction originates from paper [1] where a simpler version of this concept has been used as a module in a smart alarm clock application. During the development of the application multiple data sources were investigated, some of which turned out to be unusable. In this section the details of the selected data sources will be discussed.

1.1 Traffic

I found that the most reliable publicly available data source for traffic is the official map of Hungarian State Railways [2], where all trains can be tracked in real-time. I have created a small automated script that runs on a virtual private server and takes a snapshot of the map approximately every minute and stores the result in a JSON file.

Traffic data have been being collected since January 2019, which means there are roughly 1 year and 6 months of available information (approx. 130 million records). Due to the COVID-19 outbreak a data freeze was applied at the end of March 2020 because of the extraordinary circumstances that affect transportation all over the world. Hungarian State Railways canceled lots of trains and only 30% percent of a train's capacity can be used in order to prevent the spread of the infectious disease. This new situation significantly alters the operation of the railway system which would have introduced a lot of noise to the existing dataset and it might not be relevant in a few months at all. Should the pandemic be over its effects could be analyzed later on but currently it is out of scope of this paper.

Table 1: Details of a train entry in a snapshot

Field name	Example
Date	"2019.10.29 20:09:38"
Elvira ID	"5614115_191029"
Operator	"MAV"
Line	"40"
Train number	"55808"
Relation	"Budapest-Keleti - Pécs"
Latitude	46.26418
Longitude	18.10566
Delay	5

A snapshot of the map contains the following information about each of the trains that were present at the time the snapshot was taken (Table 1).

1.2 Weather

In addition to the traffic data we also collected the corresponding weather data for every train, because we suspect that weather has an influence on the delays as well. It was not easy to find a free provider which is capable of handling the necessary amount of requests, but after many trials we decided to use OpenWeatherMap [3]. Its free tier gives access to 60 location-based weather requests per minute, which is still not enough for every individual train, but can be sufficient to place virtual weather stations all over Hungary with a resolution of approximately 35.5 km.

Definition 1. *Virtual weather station. A virtual weather station is a GPS position which can be queried for up-to-date local weather information.*

Calculating the coordinates of the virtual weather stations The first task is to distribute the available 60 slots uniformly such that every train can be assigned to the closest virtual weather station. Finding an exact solution to the problem would have been infeasible, therefore we decided to develop an approximation algorithm for which we used the GeoNames geographical database [4] which contains POIs in Hungary and is available for download free

The algorithm (Algorithm 1) uses a k-d tree which is a space-partitioning data structure that allows fast nearest neighbor searches. [5] The k-d tree is used to place 60 virtual weather stations on the map as follows: in each step the most populated POI is selected and then its neighboring POIs are eliminated in the given `radius`. It results in an approximately uniform placement of virtual weather stations and they are located at densely populated areas where accurate weather information benefits more people.

Algorithm 1 Approximation algorithm for virtual weather station placement

Funct getVirtualWeatherStationPositions($pois, radius$)

```
pois ← pois.sort("population","desc")
kdt ← kdTree < POI > ("haversine")
for poi ∈ pois do
    nn ← kdt.searchRadius(poi, radius)
    if nn = ∅ then
        kdt.add(poi)
    end if
end for
return kdt
```

2 Analysis

2.1 Reconstruction of the railway network

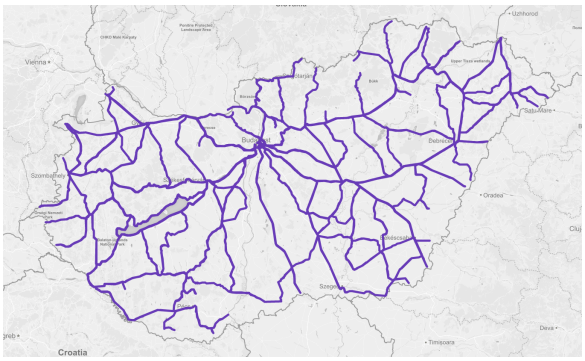
In the traffic dataset there are millions of recorded GPS coordinates and the majority of them can be safely discarded after the necessary information have been extracted. For this task we used the *Representative point extraction and updating algorithm* by Zhongyi Ni et al. [6] which is able to calculate the most significant points along a route and is also capable of refining these points as new data becomes available due to its online nature. The task is to determine the least amount of points (called the representative points) that can accurately represent such a route.

The above-mentioned algorithm can determine the points describing a route (Figure 1) with an arbitrary resolution. However, the recorded GPS trajectories are noisy and may contain significantly misplaced outliers. The representative point extraction algorithm is able to properly handle noise, but it also creates new representative points in case of outliers, therefore a support-based post-processing step is needed, which removes representative points that are encountered rarely.

2.2 Conflicting trains

As a dimensionality reduction method it is beneficial to obtain the set of trains that might affect the delay of another train. It can be also used to model delay-chain propagation.

Figure 1: Reconstructed railway network of Hungary



Definition 2. *Conflicting trains. A set of trains is said to be in conflict when their route has common representative points.*

The algorithm (Algorithm 2) determines the set of representative points which are within a given distance `radius` to a specific representative point `rp` and returns the set of trains that travel through those points without taking the temporal dimension into consideration, because we only use the intersection of conflicting trains with the currently traveling trains.

Algorithm 2 Algorithm for determining conflicting trains

Funct getConflictingTrains($allRps, rp, radius$)

```
kdt ← kdTree < RP > ("haversine", allRps)
nn ← kdt.searchRadius(rp, radius)
return ⋃_{n∈nn} {n.trainId}
```

2.3 Association rules

We realized that the grouping of trains can be considered as a frequent itemset mining problem, therefore we used the Apriori algorithm [7] for itemset mining and association rule learning.

Definition 3. *Delayed train. A train is officially considered to be delayed when its delay is greater than or equal to 5 minutes.*

The algorithm requires transactions, which can be constructed based on the snapshots of the map. For each snapshot a transaction is made based on the set of conflicting delayed trains (Definition 2) in the given snapshot.

Association rules were generated for departure delays (Table 2), for which the snapshots taken upon the scheduled departure are used. The meaning of a rule is that if the set of antecedent trains are delayed then the consequent train is likely to depart late with the given metrics.

Low support values are due to the fact that train 2749 is only included in a transaction when it is delayed. The

Table 2: A subset of association rules generated for train 2749

Antecedents	Consequent	Supp.	Conf.
2879,2739	2749	0.21	0.81
2879,7039,2739	2749	0.19	0.84
2879,2859,2739	2749	0.17	0.82
2879,7039,2859,2739	2749	0.16	0.84
2879,700,7039	2749	0.15	0.80
2879,700,2859	2749	0.15	0.83
2879,6299,2739	2749	0.14	0.87
2879,7039,6299,2739	2749	0.14	0.89
2879,700,2739	2749	0.14	0.83
700,7039,2859,2879	2749	0.13	0.85
2879,2740,2739	2749	0.12	0.93
2879,2740,2859	2749	0.12	0.84

frequent itemset containing train 2749 has a support value of 0.36 which means the train departs late roughly 36% of the time.

2.4 Sequential rules

Besides the traditional association rule mining we can also consider consecutive snapshots of a specific train on a given day, which takes the temporal information into consideration as well (Table 3). Sequential pattern mining is almost the same as association rule mining, but instead of working directly with a transaction we consider consecutive transactions recorded in time.

Sequential rules can be also mined for the departure delay, but they are more meaningful if we mine them along the entire route of the train. The rule $A \implies B$ means that when the trains in A are delayed then train B will also become delayed in the future. In case of association rules we talked about trains that are usually delayed together, but now we have an additional temporal dimension.

In order to test this method we used the SPMF open-source data mining library [8] with the RuleGrowth algorithm [9].

The support values are much higher in this case, because rules are mined along the entire route of the train. The support value of the frequent itemset containing train 2749 is 0.71 which means even though the train departed late only 36% of the time it got delayed 71% of the time during its trip.

2.5 Other factors

In addition to the delay propagation there might be other factors that contribute to the delay of trains, like weather and temporality. In this section some of these factors will be analyzed with possible explanations and conclusions.

Table 3: A subset of sequential rules generated for train 2749

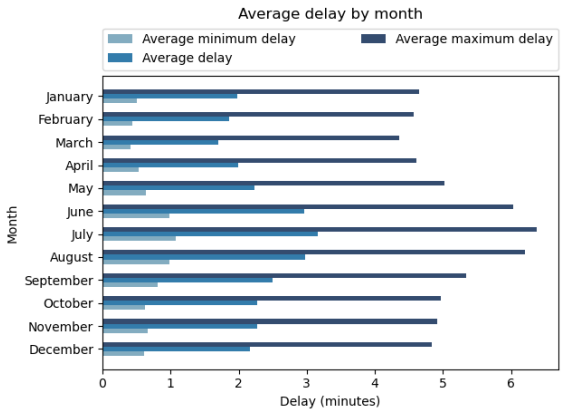
Antecedents	Consequent	Supp.	Conf.
2879	2749	0.63	0.97
7049	2749	0.61	0.97
2669,2879	2749	0.60	0.97
2669,6099	2749	0.61	0.90
2669	2749	0.67	0.90
6099	2749	0.63	0.87
2649	2749	0.61	0.86
7009	2749	0.61	0.84

Definition 4. *Average delay. The average of the trains' average delays along their route on a given day.*

Definition 5. *Average minimum (maximum) delay. The average of the trains' minimum (maximum) delays along their route on a given day.*

Month It turned out that summer is the only specific season which has a peak in the average delays followed by autumn (Figure 2). This effect might be caused by maintenance works, but there is no available historical maintenance data to confirm this theory. It's likely not caused by the number of passengers since there is no school in Hungary during the summer, which significantly reduces the number of passengers in the rush hours. Higher temperatures also seem to have an effect on the delays.

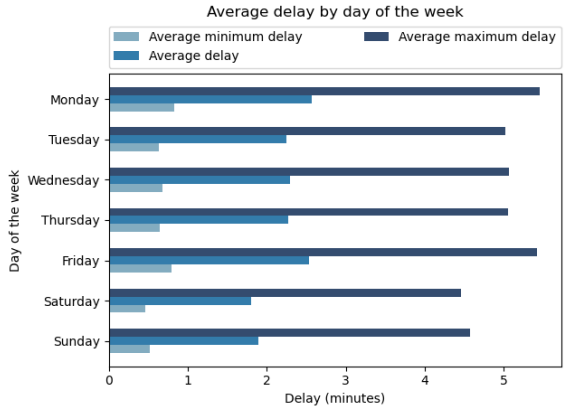
Figure 2: Average delays grouped by month



Day of the week By looking at the average of all trains we can claim that Monday and Friday have the largest delays on average, while weekends have somewhat lower average delays (Figure 3). It would be nice to have a dataset related to the number of passengers because the larger amount of

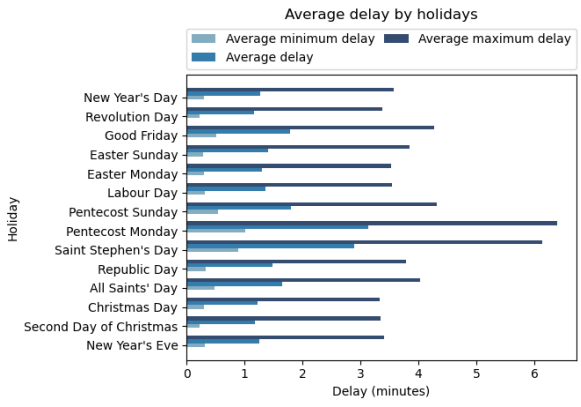
passengers may cause delay peaks at the beginning and at the end of the workweek. The number of passengers may also have a correlation with the lower delays during the weekend, but we suspect that it is likely caused by the sparser schedule, which effectively reduces delay propagation.

Figure 3: Average delays grouped by day of the week



Holidays Holidays do not seem to have a significant effect on the average delays (Figure 4). The peaks were mostly predictable according to the previous researches - Pentecost Monday and Saint Stephen's Day have slightly higher average delays but they are both in the Summer, which has the highest average delay among the seasons and Good Friday is a Friday, which has above average delay if we compare it to the other days of the week. As a conclusion, events do not seem to cause extraordinary delays, because they can be planned ahead.

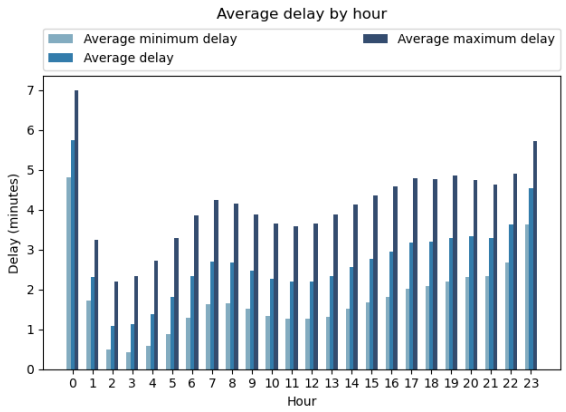
Figure 4: Average delays grouped by holidays in 2019



Time of the day By looking at the chart containing the delays grouped by hours, the rush hours can be clearly marked as delay peaks (Figure 5). It can be concluded that as the number of passengers and the density of the schedule increase, the average delay increases as well. According to the research, most relations have this pattern.

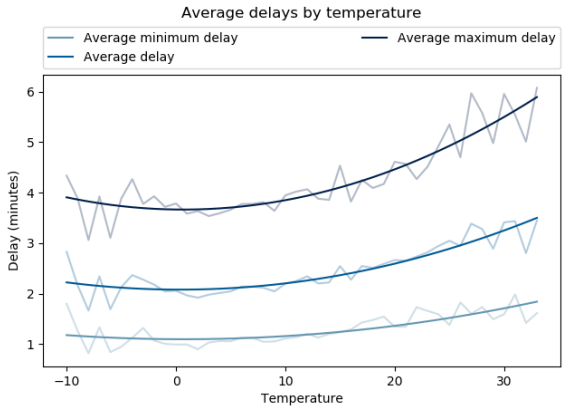
Two other peaks can be observed between 23:00 and 01:00. In order to understand them domain knowledge is needed. The reason behind the existence of the peaks is that only a very small number of train travels by that time in the country (sometimes even less than 10), and when some of them are delayed, it causes a huge impact on the average.

Figure 5: Average delays grouped by hour



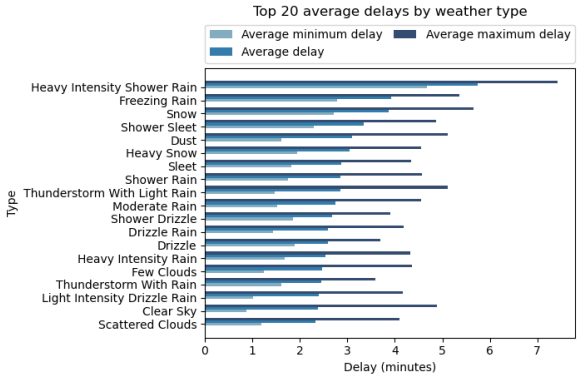
Temperature The chart shows that the average delays increase as temperature tends to either -10 or +30 Celsius degrees (Figure 6). Due to the distribution of trains, the ends of the chart are noisy, but the trendline can be easily seen.

Figure 6: Average delays grouped by temperature



Weather type As far as the type of weather is concerned, precipitation usually increases the delays (Figure 7). The most troublesome types are related to snow in the winter and unexpected thunderstorms in the summer. Nearly all rain types have higher average delays than clear sky.

Figure 7: Average delays grouped by weather type



2.6 Delay heatmap

An interesting visualization method is to generate a heatmap of delay changes (Figure 8). It allows us to see where the delay accumulates during the trip and these peaks might suggest track problems, busy stations, or any other hidden issues that we are not aware of.

Figure 8: Delay heatmap of train 2749 between Monor and Budapest-Nyugati

In this figure, red means larger average increase of delay and blue means a lower average increase of delay. The green patches represent moderate average increases of delay.

3 Departure delay prediction

Traveling in an unreliable environment on a daily basis can be nerve-wracking. The official mobile application of Hungarian State Railways has a delay forecasting mechanism, but it is quite limited in its current form. When a train is already moving then the schedule is automatically adjusted by its current delay. This estimation is not so reliable on the long term, but it can give you an idea about the scale of the expected delay under the current circumstances.

Another problem is that the forecast lacks a very important indicator, as it cannot tell whether the train is going to depart late or on-time. The forecast is only available after the train has already departed. The two main goals are to find a method to predict the departure delay and to improve the long term reliability of the delay forecast mechanism already present in the application.

Departure delay prediction is a special problem, because we do not have any information yet about the train we are interested in. Whether the train is going to depart late or on-time can only be predicted based on its observable environment. The input for the departure delay prediction problem is a set of snapshots taken at the scheduled departure time for which the target value is the delay of the train on its first appearance on the day.

3.1 Association rules

The first idea is that the previously mined association rules (Table 2) should be applied and see if we can predict whether a train is going to be delayed or not upon departure.

The algorithm (Algorithm 3) of the model is very simple, it only requires a set of association rules extracted based on the input for departure delays. A train is considered to be delayed if it is a consequent in a rule for which all the antecedent trains are delayed in a given snapshot of the map. The hyper-parameters of the model are the minimum support and minimum confidence of the rules.

Algorithm 3 Algorithm for predicting the departure delay (association rules)

Funct predictDepartureDelayAr($snapshot, rules$)

```
delayedTrains ← getDelayedTrains(snapshot)
for rule ∈ rules do
    if rule.getAntecedents() ⊆ delayedTrains then
        return True
    end if
end for
return False
```

The results (Table 4) are impressive, but according to the research it turned out there are simply not enough information for the association rule mining algorithm in its current form which causes underfitting. Trains are categorized as either delayed or on-time, which cannot properly handle the following situation: when an antecedent train is delayed more than a given threshold (for example 10 minutes) then the consequent train can depart on-time as they are far away from each other and a slot becomes available for the consequent train. Otherwise, the delay of the antecedent train propagates to the consequent train.

Table 4: Departure delay prediction metrics for train 2749 using the association rules on the test set

	Precision	Recall	F1-score	Support
On time	0.85	0.85	0.85	194
Late	0.72	0.71	0.72	105
Accuracy			0.80	299

3.2 Train embedding

In order to solve the underfitting problem that affects the association and sequential rules, a different approach is necessary. It is not enough to have an indicator whether a train is delayed or not, the exact numeric values are needed instead. It is also important to have an input with fixed length for the algorithms.

The solution (Algorithm 4, Table 5) is that each conflicting train that travels when a specific train departs is considered as a unique feature with its current delay. If a previously encountered conflicting train is not present at the time, its delay becomes 0 as it likely won't affect the delay-chain. First, the algorithm is called with an empty state vector and a subset of trains from a snapshot. If the identifier of a train is not contained in the state vector then it is appended to it. For each train identifier in the state we determine whether it is present in the current input or not and we append its current delay to the embedding. If a train is not found in the input, its current delay is considered to be 0. The returned state can be then re-used to embed another set of trains. Before training, the embeddings can be safely padded with zeros to have a common length.

Algorithm 4 Algorithm for creating train embeddings

Funct embed($state$,$trains$)

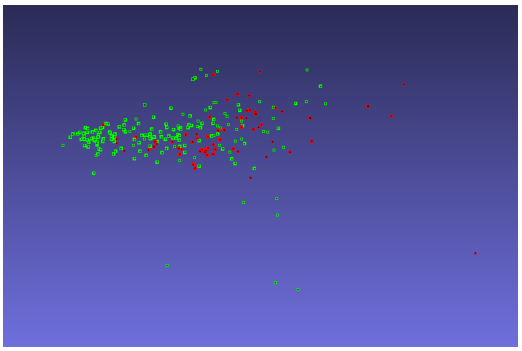
```
1:  embedding ← []
2:  for train ∈ trains do
3:      if train.getId() ∉ state then
4:          state.append(train.getId())
5:      end if
6:  end for
7:  for trainId ∈ state do
8:      if trainId ∈ trains then
9:          embedding.append(trains.getDelay(trainId))
10:     else
11:         embedding.append(0)
12:     end if
13: end for
14: return state, embedding
```

They usually have high dimensions but they can be visualized using dimensionality reduction methods (Figure 9). In the following picture trains that depart on-time are denoted by green squares and trains that depart late are marked as red squares.

Table 5: An example train embedding

406	472	580	609	619	709	2617	...
3	18	1	3	3	0	2	...
3	20	2	3	4	0	1	...

Figure 9: Embeddings for train 2749 visualized using 3-dimensional PCA



3.3 Support-vector machine

A support-vector machine [10] tries to find a hyperplane in an n-dimensional space which separates, and therefore classifies the data points. This hyperplane should have maximum margin, which means it should have maximal distance between the two classes, so future data points can be classified more reliably.

For each train, a unique model is trained. In our example, the space is 45-dimensional and the hyperplane separates the trains that depart on-time and the trains that depart late. For the SVM experiment we've implemented a grid-search (Table 6) and executed it on the training set with 5-fold cross-validation with the following parameters:

Table 6: Parameter grid for the SVM experiment

Parameter name	Possible values
Regul. parameter (C)	0.001, 0.01, 0.1, 1, 10
Kernel	linear, poly, rbf, sigmoid
Kernel coeff. (gamma)	0.001, 0.01, 0.1, 1
Indep. term (coef0)	0.0, 0.001, 0.01, 0.1, 1, 10

The grid-search optimizes the hyper-parameters of the SVM model on the training dataset which results in better

metrics during the evaluation phase. The best parameters were C=1, coef0=10, gamma=0.01 and kernel=poly. (Table 7)

Table 7: SVM prediction metrics for train 2749

	Precision	Recall	F1-score	Support
On time	0.92	0.97	0.94	194
Late	0.94	0.84	0.88	105
Accuracy			0.92	299

3.4 Random Forest Classifier

A random forest [11] is an ensemble model which fits multiple decision trees and outputs their mode. Each decision tree splits the dataset a variable number of times based on the delays of the conflicting trains and outputs whether the train is going to depart late or not.

The training methodology was similar to SVM's, we ran a grid-search (Table 8) with 5-fold cross-validation on the training set with the following parameters:

Table 8: Parameter grid for the Random Forest Classifier

Parameter name	Possible values
n_estimators	200, 600, 1200, 1800
max_depth	10, 50, 100, unlimited
min_samples_split	2, 5, 10
min_samples_leaf	1, 2, 4
bootstrap	True, False

The results (Table 9) are slightly better than the SVM's, and the trained model also helps with the explainability of the delay-chains, which is useful to prevent them from occurring in the future. The best parameters were bootstrap=true, max_depth=10, min_samples_leaf=2, min_samples_split=10 and n_estimators=200.

Table 9: Random Forest Classifier prediction metrics for train 2749

	Precision	Recall	F1-score	Support
On time	0.97	1.00	0.98	194
Late	1.00	0.90	0.95	105
Accuracy			0.97	299

4 Generic delay prediction

Based on the research experiences with departure delays it is time to solve the prediction problem in general with deep neural networks. As it was mentioned before, the delay estimation in the official mobile application is not so reliable on the long term, therefore it would be beneficial to find a method for predicting the real schedule. The reason behind choosing deep neural networks is that state-of-the-art multivariate time series forecasting methods tend to use these technologies. [12]

The general delay prediction task will be formulated as a regression problem instead of classification, because it is more informative for the end-user and there are significantly more data available when we consider snapshots after departure as well.

4.1 Input

For each day the snapshots containing a given train t are collected in ascending order by their timestamp (Table 10). It must be noted that there can be a different amount of snapshots for each day, because a delayed train obviously travels for a longer period of time. In this section a daily collection of ordered snapshots for a given train t will be referred as the input.

Table 10: A subset of the input for train 2749 on a given day

Date	Train	Delay	Lat	Lon	...
19-06-16 06:39	2749	0	47.35	19.43	...
19-06-16 06:40	2749	0	47.35	19.43	...
19-06-16 06:40	2749	0	47.35	19.42	...
19-06-16 06:41	2749	1	47.36	19.42	...

Based on the data, there are two kinds of preprocessed inputs for each day, a vector of auxiliary features and a matrix of time-series features (Table 11).

The auxiliary features include an indicator whether the given day is a weekday, an indicator whether the given train departs during the rush hours and the one-hot encoded representation of the month upon departure.

For the time-series features a similar train embedding is used as before, the only difference is that this time the train we are interested in is also included in the embedding. The embedding has a much higher dimensionality, because conflicting trains are embedded over the entire route of the train. In order to keep the input dimension manageable, only the characteristics of the train embeddings are used.

An entry in the time-series input contains the current delay of the train, the classified weather and the mean, standard deviation, minimum and maximum values of the delays of the conflicting trains.

Let's suppose that train t traveled during k days over the interval covered by the dataset and the number of time-series features for all of its snapshots are m. Thus the 3D

Table 11: Preprocessed time-series input on a given day

Delay	Weather	Mean	STD	Min	Max
0	3	1.20	2.69	0	13
0	3	1.25	2.65	0	13
0	3	1.32	2.62	0	13
1	3	1.35	2.60	0	13

input of the network has dimensions (k, l_i, m) where l_i is the number of snapshots on the i^{th} day.

4.2 Output

For each entry in the time-series input the corresponding output becomes the vector of true delays in the future after n minutes where $n \in \{5, 10, 20, 30\}$. Let's assume that the train we are interested in is t and its current delay is determined by $d_t(X_i)$. For each snapshot X_i let the output y_{ij} equal to the delay of train t at snapshots X_{i+n_j} $(j = 1..4)$. In case of out of bounds indices the delay of t at the last snapshot of the day is used instead.

$$y_i^{true} = [d_t(X_{i+5}), d_t(X_{i+10}), d_t(X_{i+20}), d_t(X_{i+30})]$$

Official model The main goal of the generic delay prediction task is to obtain a more accurate forecast than it is currently available in the official mobile application. In order to have a meaningful comparison, we have to recreate the model of the official forecast method and calculate its loss and other metrics alongside our model. Fortunately, the official model is not too complicated, it simply substitutes the current delay for all future occurrences.

$$y_i^{official} = [d_t(X_i), d_t(X_i), d_t(X_i), d_t(X_i)]$$

LSTM model Our model has to support both the auxiliary and time-series features, therefore a multi-input network is necessary. This problem is similar to the image captioning task, where an image is chosen as an auxiliary feature and the words of the generated caption are sequence-like. [13, 14] Due to the fact that there can be a varying number of snapshots per day some sort of recurrent neural network (RNN) is needed, which can handle the temporal nature of the data as well. The output of an RNN depends not only on the current input but on the previous outputs as well. Its memory is very useful for the prediction of the delays, because it can learn complicated delay patterns.

The RNN can also have a preset initial state, where we can store the representation of the auxiliary features and the resulting network models $P(X_{i+1}|X_{0:i}, auxiliary)$. [15] This auxiliary condition allows us to have a single network for all trains if we include a train identifier, but due to resource constraints this was not used during the research.

Figure 10: Comparison of different 10-minute prediction models for train 2749 on a given day

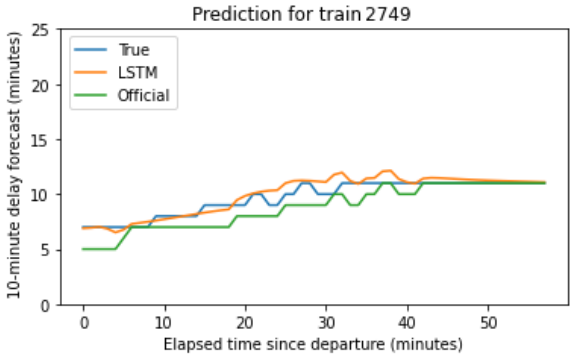


Table 12: Evaluation of the generic prediction model on train 2749

	n = 5	n = 10	n = 20	n = 30
LSTM MSE	1.03	2.01	4.23	7.07
LSTM R^2	0.97	0.95	0.91	0.84
Official MSE	1.28	3.03	7.63	12.92
Official R^2	0.97	0.93	0.83	0.72

4.3 Evaluation

The first evaluation was performed on train 2749. Out of the 236 available occurrences only 231 were used, where the maximum delay along the route was less than 30 minutes. There is simply not enough data for the outliers where delay may occasionally exceed 250 minutes.

The following metrics (Figure 10, Table 12) were calculated using 3-fold cross-validation.

For this train the LSTM model gives better and better results as n increases compared to the official model.

On average, the proposed LSTM model outperforms the official model, but only when significant outliers are omitted from the dataset. The proposed model is not able to learn extreme delays yet reliably due to their rare nature, but the official model is able to forecast them easily by simply substituting the current delay in a linear manner. This is not a huge issue, because if a train is delayed that much it usually skips its trip on that day entirely and passengers are informed on multiple platforms.

4.4 Conclusion

The analysis and the machine learning models presented in this paper could be useful for the betterment of railway services in Hungary and they may also increase the satisfaction of the passengers. Hungarian State Railways also expressed their interest in the continuation of the research project in cooperation with our university.

References

[1] Roland Krisztián Szabó. Smart alarm clock based on traffic and weather information, 2018.

[2] MÁV Szolgáltató Központ Zrt. MÁV-START térkép, 2020. [Online; accessed 16-January-2020].

[3] Openweather Ltd. OpenWeatherMap, 2020. [Online; accessed 16-January-2020].

[4] GeoNames Team. GeoNames dump (Hungary), 2020. [Online; accessed 16-January-2020].

[5] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, September 1975.

[6] Zhongyi Ni, Lijun Xie, Tian Xie, Binhua Shi, and Yao Zheng. Incremental road network generation based on vehicle trajectories. *ISPRS International Journal of Geo-Information*, 7(10), 2018.

[7] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[8] Philippe Fournier-Viger. SPMF open-source data mining library, 2020. [Online; accessed 17-March-2020].

[9] Philippe Fournier-Viger, Roger Nkambou, and Vincent Shin-Mu Tseng. Rulegrowth: Mining sequential rules common to several sequences by pattern-growth. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, page 956–961, New York, NY, USA, 2011. Association for Computing Machinery.

[10] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, September 1995.

[11] Tin Kam Ho. Random decision forests. In *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1*, ICDAR '95, page 278, USA, 1995. IEEE Computer Society.

[12] Papers with Code. Multivariate Time Series Forecasting, 2020. [Online; accessed 08-May-2020].

[13] Andrej Karpathy and Fei-Fei Li. Deep visual-semantic alignments for generating image descriptions. *CoRR*, abs/1412.2306, 2014.

[14] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. *CoRR*, abs/1411.4555, 2014.

[15] Philippe Rémy. Conditional RNN, 2019. [Online; accessed 17-March-2020].

EFOP-3.6.3-VEKOP-16-2017-00001: Talent Management in Autonomous Vehicle Control Technologies – The Project is supported by the Hungarian Government and co-financed by the European Social Fund.